|  |
| --- |
| Abstract |
| Heart disease is the leading cause of death in the world, claiming 1 in 3 deaths globally. Physicians run several tests to identify whether a patient is at risk, but it’s hard to rely on the result of a single test to make their final decision. Instead, it’s a combination of several factors, and sometimes it may be hard to pinpoint the potential risk among too many parameters. My objective is to develop a web application for physicians to predict a patient’s heart disease diagnosis by running a machine learning algorithm behind the scenes. Medical personnel can also add new patient data into the model to improve its accuracy over time. The tool can be used on any computer or mobile device. |

|  |
| --- |
| Introduction |
| Cardiovascular diseases (CVDs) kill an American every 34 seconds, and half of the population has some form of cardiovascular issue (Centers for Disease Control and Prevention, 2017). Physicians run several tests to diagnose coronary heart disease, which range from echocardiogram to stress test, from CT scan to MRI (National Institute of Health, 2018). However, it may turn into a daunting task to figure out whether the patient has a potential for high risk among all those factors. Machine learning can be of great help to build a model from existing patient data and to predict the health status of new patients by using that model.  Machine learning is the scientific study of algorithms and statistical models that computer systems use to effectively perform a specific task without using explicit instructions, relying on patterns and inference instead (Wikipedia, 2019). Machine learning problems can be grouped into classification and regression. Classification is the task of predicting a discrete class label, while regression is the task of predicting a continuous quantity.  Machine learning methods can be grouped by the learning type. In supervised learning, all data is labeled, and the algorithms learn to predict the output from the input data. In unsupervised learning, all data is unlabeled, and the algorithms learn to inherent structure from the input data by clustering and association (Machine Learning Mastery, 2019).  Machine learning algorithms, also known as classifiers, can be grouped in terms of their functionality. Ensemble algorithms are models composed of multiple weaker models that are independently trained and whose predictions are combined to make the overall prediction. When building a machine learning model, one of the most important steps in the process is evaluating the model’s performance (Towards Data Science, 2018).  Confusion matrix shows the predicted vs. actual classification:   |  |  |  | | --- | --- | --- | |  | **Negative (predicted)** | **Positive (predicted)** | | **Negative (actual)** | true negatives (tn) | false positives (fp) | | **Positive (actual)** | false negatives (fn) | true positives (tp) |       Balanced accuracy in PMLB computes each class' accuracy on a per-class basis using a one-vs-rest encoding, then computes an unweighted average of the class accuracies (Olson, La Cava, Orzechowski, Urbanowicz, and Moore, 2017). |
| Materials & Methods |
| |  |  | | --- | --- | | * MacBook Pro by Apple | * Anaconda Python 3.7 | | * Heart Disease Data from the UCI ML Repository | * Python Flask 1.0.2 | | * Weka 3.8.3 by the University of Waikato, New Zealand | * Scikit-learn 0.20.1 | | * Penn Machine Learning Benchmarks (PMLB) Tools | * Visual Studio Code 1.30.2 by Microsoft | |
|  |
| * Downloaded the Cleveland heart disease dataset from the UCI ML Repository * Read the Penn Machine Learning Benchmark (PMLB) research paper * Installed Weka to examine the heart disease data in a graphical user interface * Converted the PMLB heart disease dataset from “tsv” into the “arff” format of Weka * Used Weka for feature selection to identify the most important attributes * Created a new heart disease dataset with a subset of the original attributes * Ran the PMLB toolset to evaluate the performance of several ML algorithms and parameters * Installed Anaconda3 that contained Python, Visual Studio Code, Flask, and Scikit-learn * Developed my application in Python to build a machine learning model * Created a web page in HTML5 and JavaScript to interact with the physician * Developed a web service in Python Flask to communicate between the web page on the client machine and the machine learning application on the server |

|  |
| --- |
| Results |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Feature Selection**          **Algorithm Selection**   |  |  |  |  | | --- | --- | --- | --- | |  | **Gradient Tree Boosting** | **XGBoost** | **Random Forest** | | **Accuracy** | 84.49% | 86.14% | 85.15% | | **F1 Score** | 84.18% | 85.91% | 84.86% | | **Balanced Accuracy** | 83.92% | 85.67% | 84.58% |     **Parameter Selection**   |  |  |  |  | | --- | --- | --- | --- | |  | **Parameter Settings** | **Evaluations** | **Time to Run** | | **Gradient Tree Boosting** | n\_estimators\_values = [10, 50, 100, 500]  min\_impurity\_decrease\_values = np.arange(0., 0.005, 0.00025)  max\_features\_values = [0.1, 0.25, 0.5, 0.75, 'sqrt', 'log2', None]  learning\_rate\_values = [0.01, 0.1, 0.5, 1.0, 10.0, 50.0, 100.0]  loss\_values = ['deviance', 'exponential'] | **7,840** (4x20x7x7x2) | **1:58:01** (0.90 sec/eval) | | **XGBoost** | n\_estimators\_values = [10, 50, 100, 500]  learning\_rate\_values = [0.01, 0.1, 0.5, 1.0, 10.0, 50.0, 100.0]  gamma\_values = np.arange(0., 0.51, 0.05)  max\_depth\_values = [1, 2, 3, 4, 5, 10, 20, 50, None]  subsample\_values = np.arange(0.0, 1.01, 0.1) | **25,200** (4x7x10x9x10) | **4:04:27** (0.58 sec/eval) | | **Random Forest** | n\_estimators\_values = [10, 50, 100, 500]  min\_impurity\_decrease\_values = np.arange(0., 0.005, 0.00025)  max\_features\_values = [0.1, 0.25, 0.5, 0.75, 'sqrt', 'log2', None]  criterion\_values = ['gini', 'entropy'] | **1,120** (4x20x7x2) | **41:13** (2.21 sec/eval) | | **Totals** | | **34,160** | **6:43:41** |  |  |  |  |  | | --- | --- | --- | --- | |  | **Gradient Tree Boosting** | **XGBoost** | **Random Forest** | | **Parameter Settings** | n\_estimators=10  min\_imp\_dec=0.00025  max\_features=0.1  learning\_rate=0.5  loss=exponential | n\_estimators=50  learning\_rate=0.1  gamma=0.4  max\_depth=3  subsample=0.2 | n\_estimators=100  min\_imp\_dec=0.00325  max\_features=sqrt  criterion=gini |     **Application Design**      **Application Development**    **Application on Devices** |
| Discussion |
| The original heart disease dataset from the UCI ML Repository had 13 attributes for each patient record. I used a feature selection process in Weka to pick 7 attributes and created a new dataset. Even though it didn’t improve the accuracy score significantly, it provided a smaller set of relevant attributes only, and simplified the user interface for the medical personnel to enter new patient data.  The Penn Machine Learning Benchmark (PMLB) research paper analyzed 13 machine learning algorithms on 165 classification datasets with 5.5 million parameter evaluations. Tree-based ensemble algorithms such as Gradient Tree Boosting, and Random Forest performed as the top two algorithms on most of the datasets.  When I downloaded the PMLB toolset, I noticed the XGBoost classifier as the 14th algorithm even though it wasn’t mentioned in their research paper. I ran the PMLB evaluation tools with 10-fold cross validation to compare the performance of Gradient Tree Boosting, XGBoost, and Random Forest on my new dataset.  Even though the results were very close to each other, XGBoost was the top performer with an 86% balanced accuracy score. It was also the fastest algorithm that completed each evaluation in 0.58 seconds. So, I picked XGBoost as the classifier with the identified parameter settings to train my model.  First, I developed the machine learning program in Python by utilizing the PMLB toolset. I was able to diagnose a new patient as healthy or sick based on the model generated by the training data. Since it wouldn’t be practical to distribute such an application to people in the medical field, I decided to turn it into a web application for anyone to run on any device.  Finally, I built a web application in HTML and JavaScript to interact with the physician and send that information to the machine learning program through a web service. The user can enter a new patient’s heart data and click a button to predict whether the patient is healthy or sick. The physician can correct the prediction with the actual health status of the patient and add the new record to the model to improve its performance as it grows with additional data over time. |

|  |
| --- |
| Conclusion |
| Bioinformatics is an exponentially growing field that combines the power of math and science through machine learning. There are many biomedical classification problems including disease diagnosis, which rely on machine learning techniques.  Expert advice was the key to success during the process of feature selection. Since the data was considerably old as being from 1990, some of the attributes selected at the time were not relevant anymore based on the latest developments in cardiology. Dr. Nafiz Kiciman, cardiologist at UCI, noticed those features even before I ran the feature selection tool in Weka. The tool just confirmed the suggestions and proved that it was a good dataset to identify irrelevant attributes successfully.  PMLB research paper and toolset helped me to compare the performance of algorithms on the new dataset after feature selection. I strongly recommend other data scientists to read the findings in this paper and utilize the provided evaluation tools on their own datasets. This approach also gave me more time to work on application development as their tools helped me to speed up the comparison process.  Developing the machine learning program in Python by using the XGBoost classifier on the new dataset was a straightforward task as there were readily available tools such as PMLB and libraries such as scikit-learn. However, letting other users connect to this program through the web was challenging. It required the development of a web page, and a web service layer to communicate with the machine learning program in the back-end.  This project provides the basic framework to run a machine learning program on a server and to communicate with it from a web page through a web service in the middle. Even though it is specific to the heart disease dataset, it can easily be modified to handle any dataset by replacing the training file and the accompanying user interface items. Please feel free to run the application on any computer or mobile device at <http://heartdx.weebly.com>. |

|  |
| --- |
| Acknowledgments |
| I’d like to thank *Nafiz Kiciman, MD, Division Chief of Pediatric Cardiology Faculty at UCI*, for his expert advice. I also thank *Ms. Cheryl Johnson*, my science teacher, for reviewing my work. Finally, I thank my father for helping me deploy the application on a public server. |